Mixed Mode Programming on HPCx

Michał Piotrowski

EPCC, University of Edinburgh

mpiotrow@epcc.ed.ac.uk

Motivation

- Clusters of shared memory nodes have become a system of choice for many research and enterprise projects.
- The mixed mode programming is a combination of shared and distributed programming models
- It can potentially exploit features of the SMP cluster architecture, resulting in a more efficient parallelisation strategy

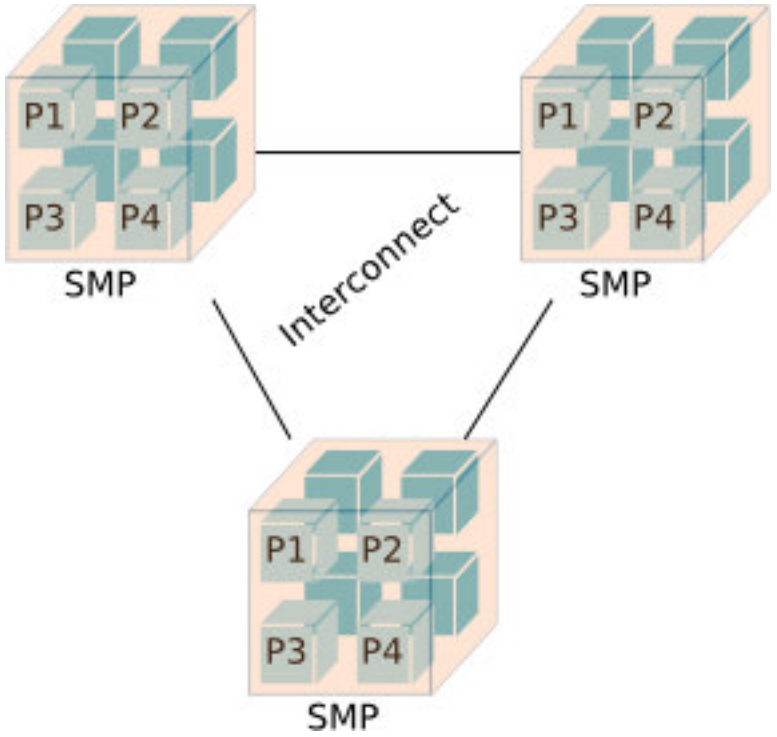


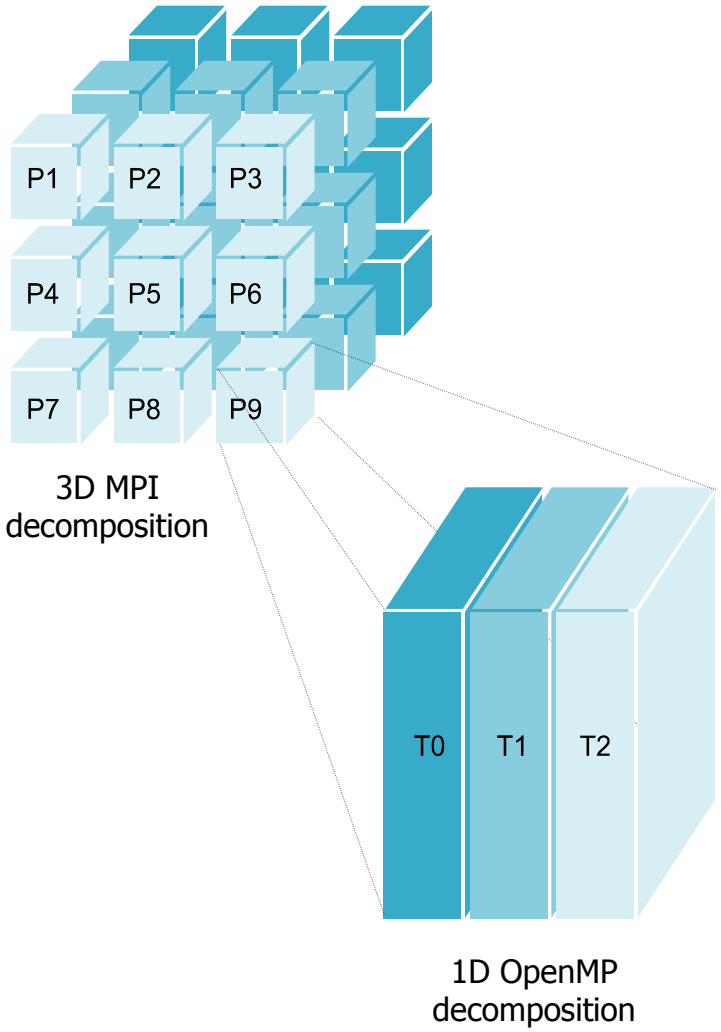
Fig 1. SMP cluster model.

Mixed Mode

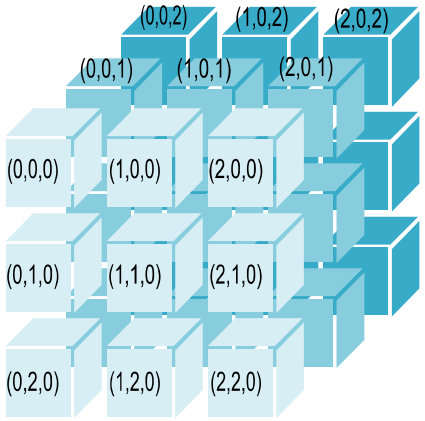
Mixed mode naturally matches the SMP cluster architecture.

Message exchange within a node is reduced and replaced by faster direct reads and writes from memory.

Typically, the number of inter-node messages will be reduced, but the average message size will increase.



Benchmark Code



while Δ is greater than tolerance

exchange halos

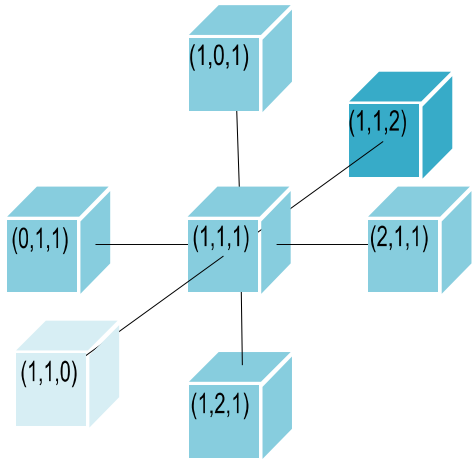
perform Jacobi relaxation

calculate global Δ

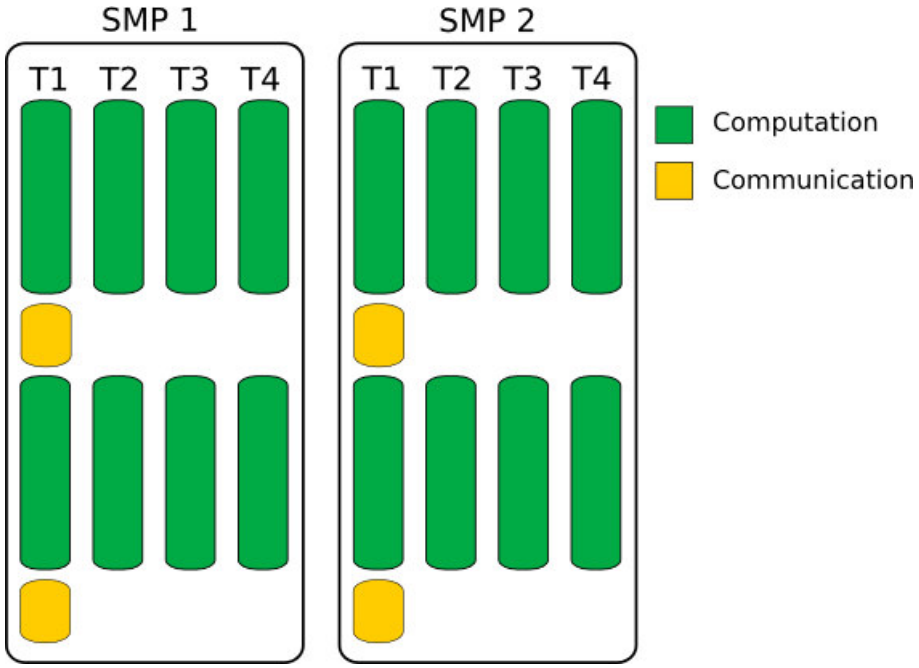
end while

Benchmark codes are based on the 3D Jacobi Relaxation Algorithm.

Benchmark code involves:
-3D data decomposition
-Jacobi computation
-Point-to-Point communication (nearest neighbours)
-Global reduction (Delta)



Master Only



Master Only: all MPI communication takes place outside of OpenMP parallel regions.

distribute data between threads

while Δ is greater than tolerance

if my thread id == 0 **then**

exchange halos

end if

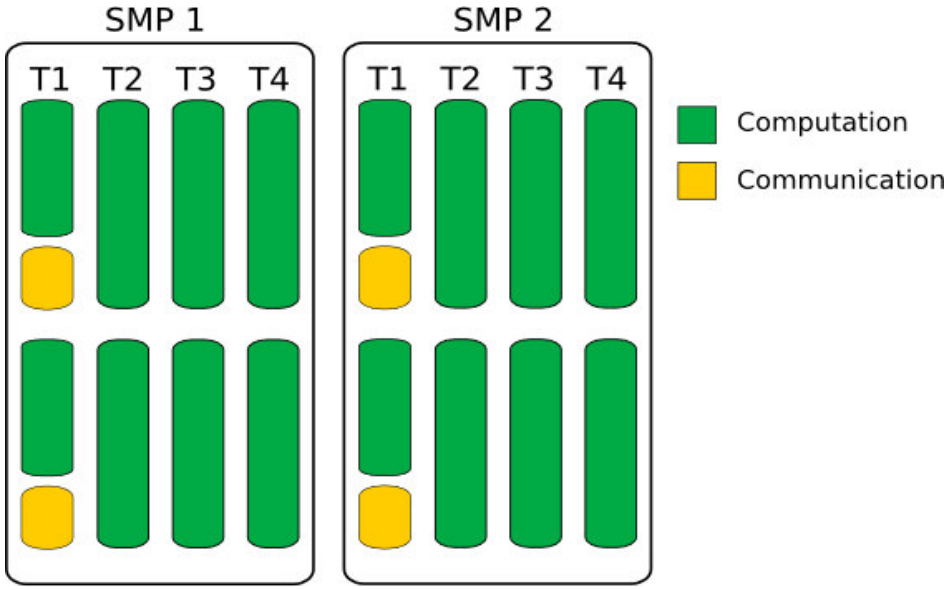
perform Jacobi relaxation

OpenMP barrier

calculate global Δ

end while

Funnelled



Funnelled: communication may occur inside parallel regions, but is restricted to a single thread.

distribute data between threads (load balance)

while Δ is greater than tolerance

if my thread id == 0 **then**

exchange halos

perform Jacobi relaxation (balanced)

else

perform Jacobi relaxation (balanced)

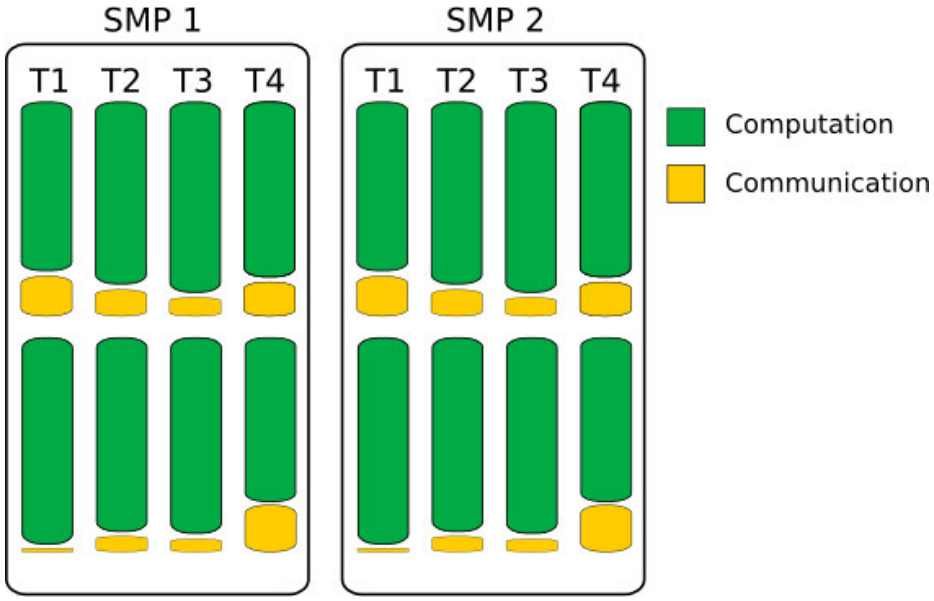
end if

OpenMP barrier

calculate global Δ

end while

Multiple



Multiple: all threads participate in the communication, independently of other threads

distribute data between threads

while is greater than tolerance

exchange 'top', 'bottom', 'front' and 'back' halos

if my thread id == 0 **then**

exchange 'left' halo

else if my thread id == max thread id

exchange 'right' halo

end if

perform Jacobi relaxation

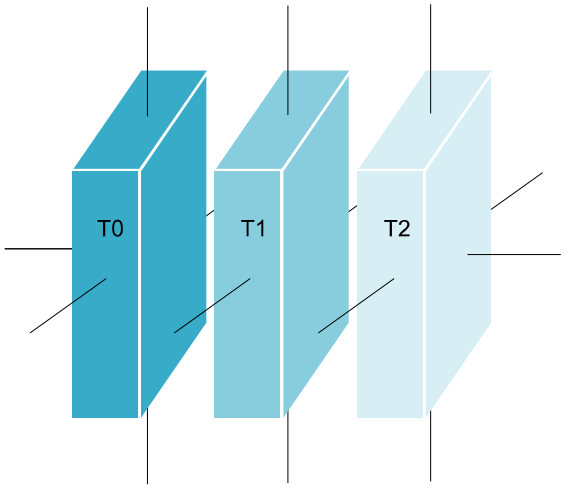
OpenMP barrier

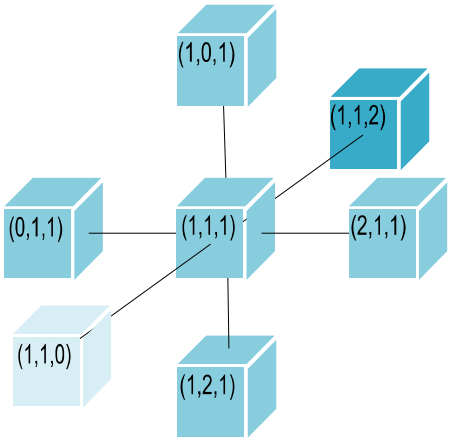
calculate global Δ

end while

Multiple - Communication

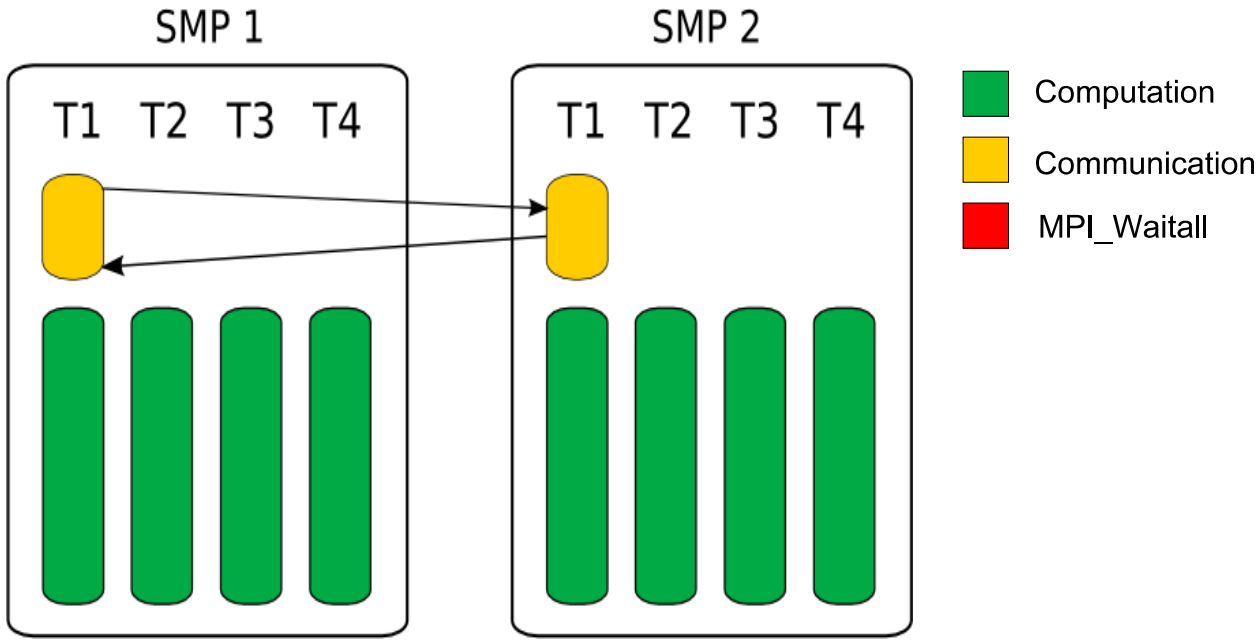
In order to recognise which message is send by which thread, tagging system is used.





A tag is an unique integer value which is based on the thread id and direction in which the message is being sent.

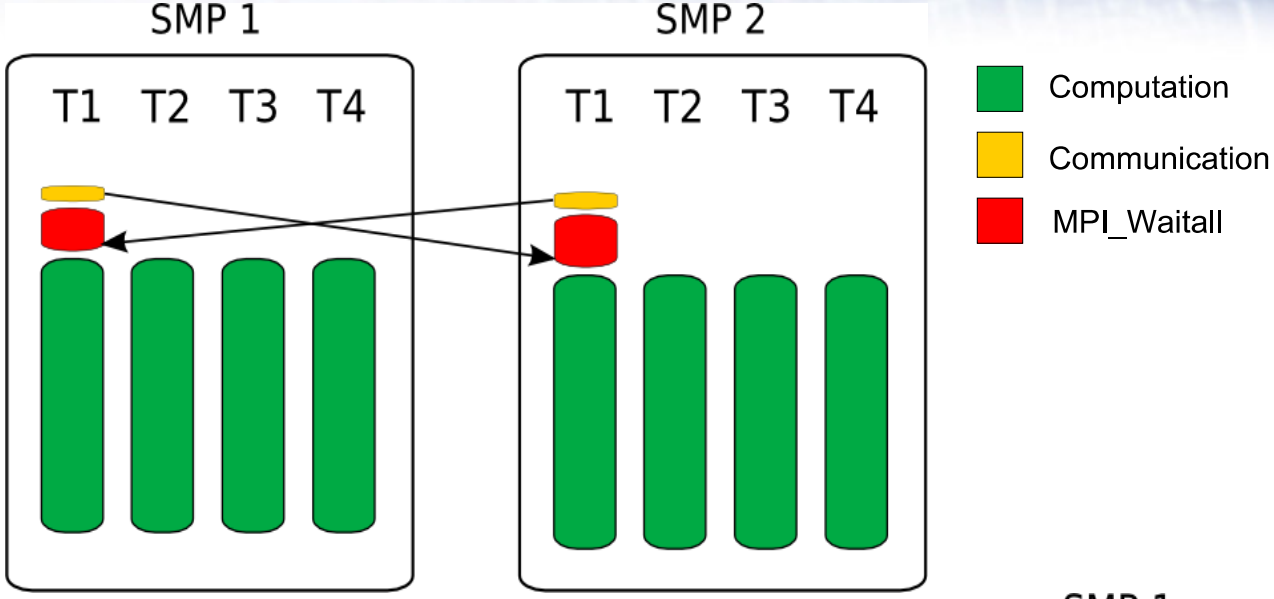
Communication - blocking



Communication - blocking

A blocking MPI call means that the program execution will be suspended until the message buffer is safe to use.

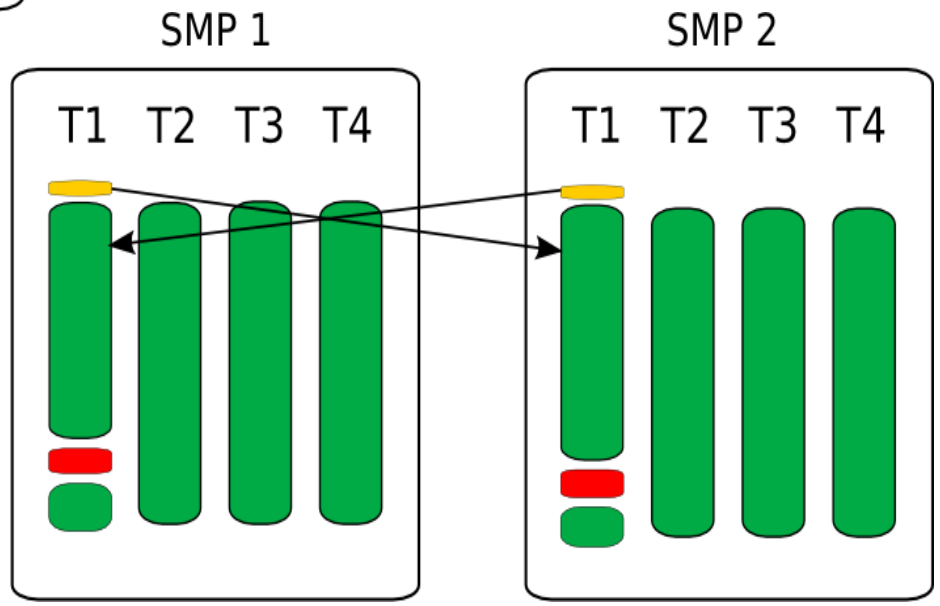
Communication – nonblocking



Communication - nonblocking

An MPI non-blocking call returns immediately after the call is initiated.

By moving the communication to the background the processors can use the former waiting time for computation.



Communication - overlapped

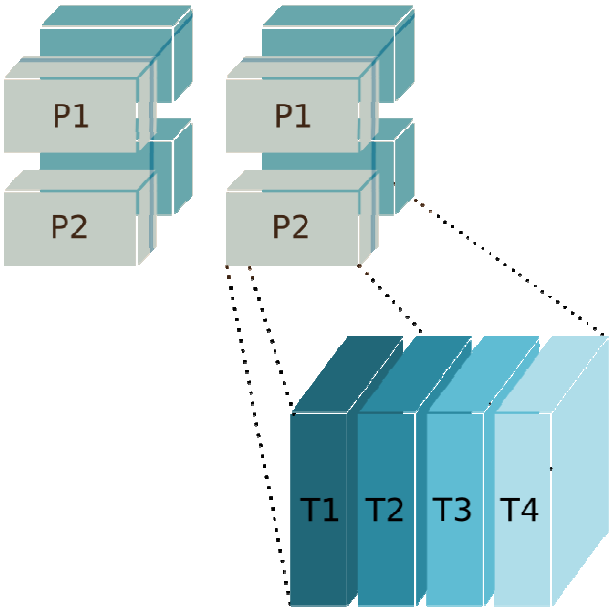
HPCx Phase 3

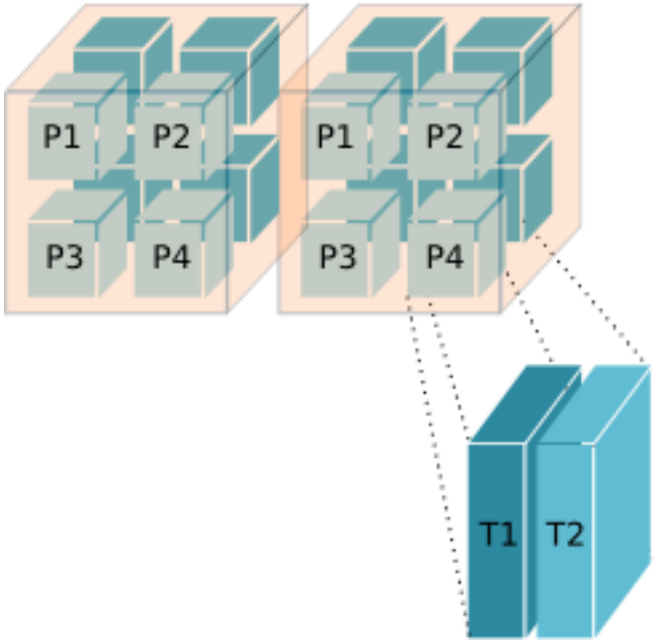
System size	160 IBM eServer 575 nodes
Node size	16 processors, 32 Gbytes of main memory
Total	2560 processors
Processor	IBM Power5 1.5GHz 64-bit RISC
Cache:	
L1	64KB instruction, 32 KB data
L2	1.9MB shared between 2 proc
L3	36MB shared between 2 proc
Interconnect	IBM High Performance Switch (HPS)

HPCx system features

Fig. HPCx System

Problem distribution



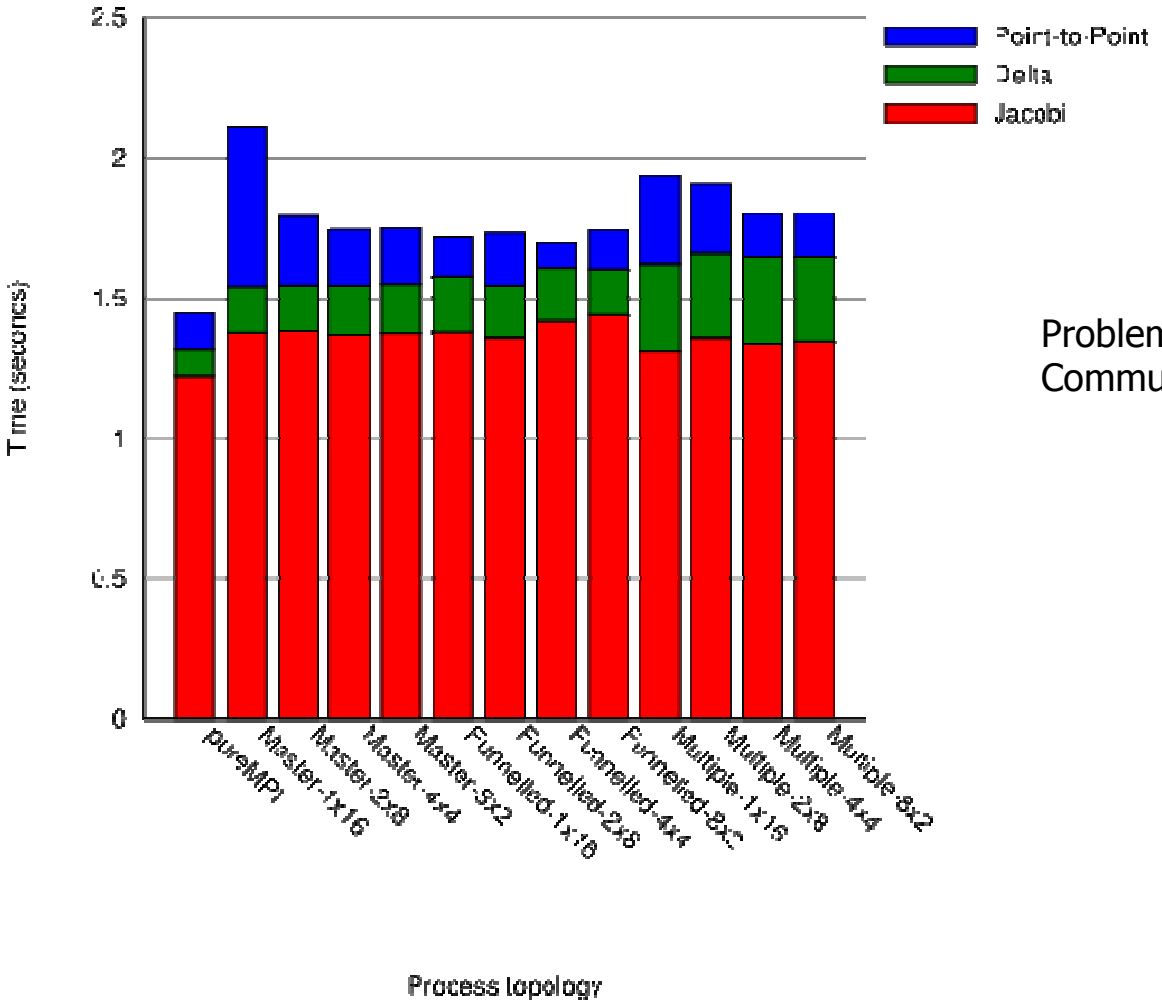


Number of processors is fixed:
256 processors, that is 16 SMP nodes.

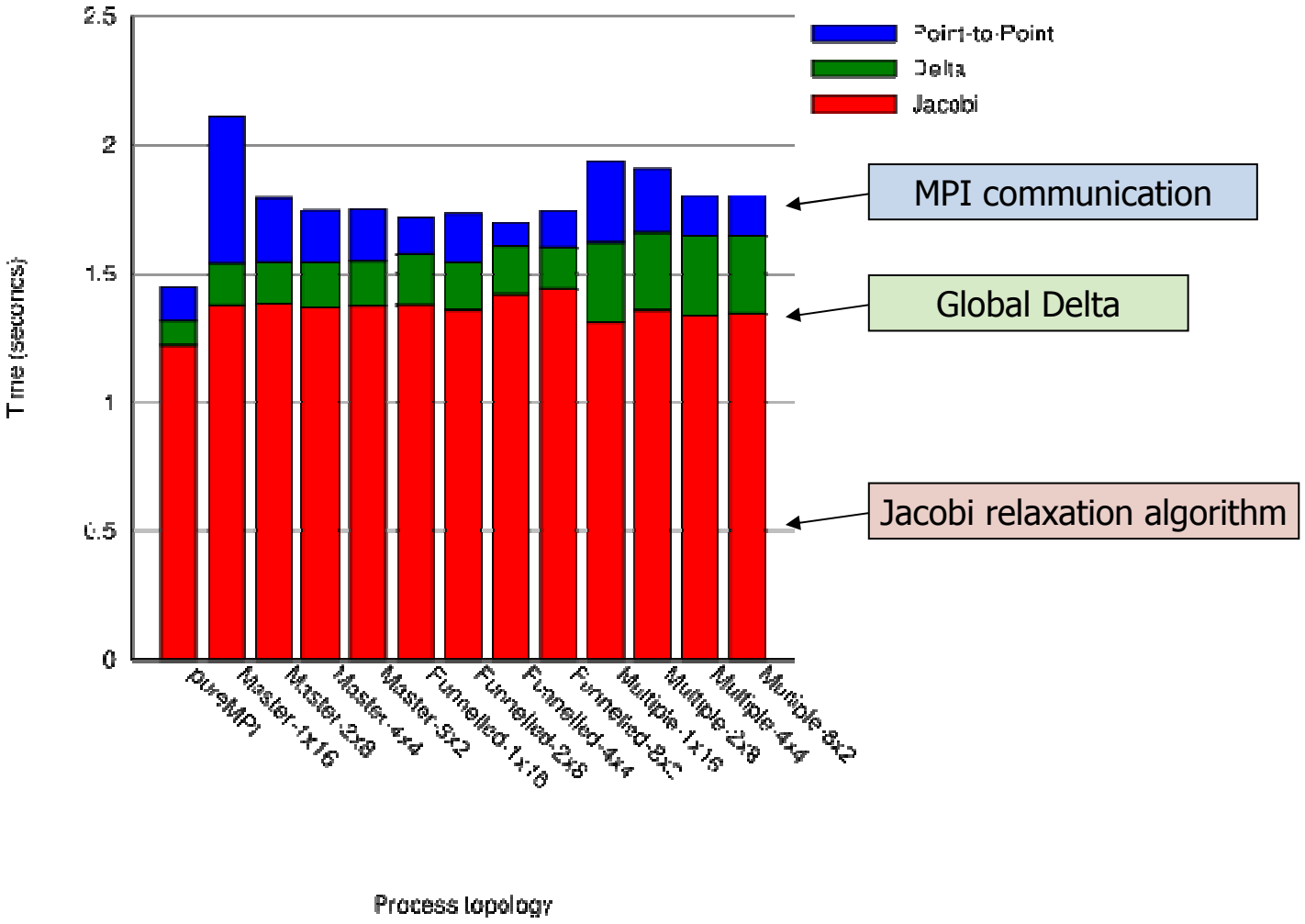
Problem size is fixed per processor:
- 96x96x96
- 192x192x192 array of doubles

independent of the balance between number of threads and processes

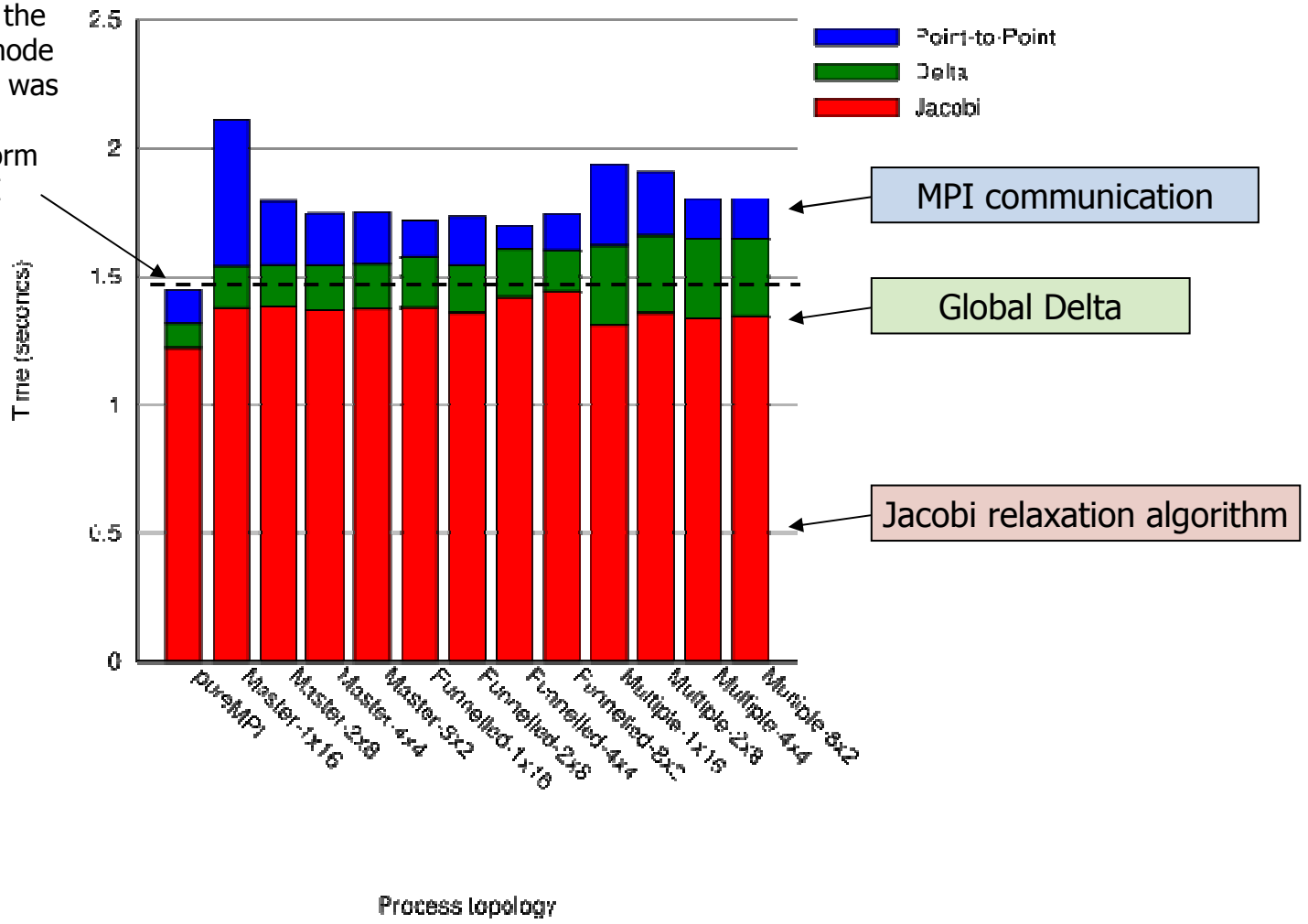
Factors connected with domain decomposition shape are excluded.



Problem size:192x192x192
Communication: blocking


2.5
2
1.5
1
0.5
0
Time (seconds)
Point-to-Point
Delta
Jacobi
MPI communication
Global Delta
Jacobi relaxation algorithm
pureMPI
Master-1x16
Master-2x8
Master-4x4
Master-8x2
Funnelled-1x16
Funnelled-2x8
Funnelled-4x4
Funnelled-8x2
Multiple-1x16
Multiple-2x8
Multiple-4x4
Multiple-8x2
Process topology

None of the mixed-mode versions was able to outperform pureMPI

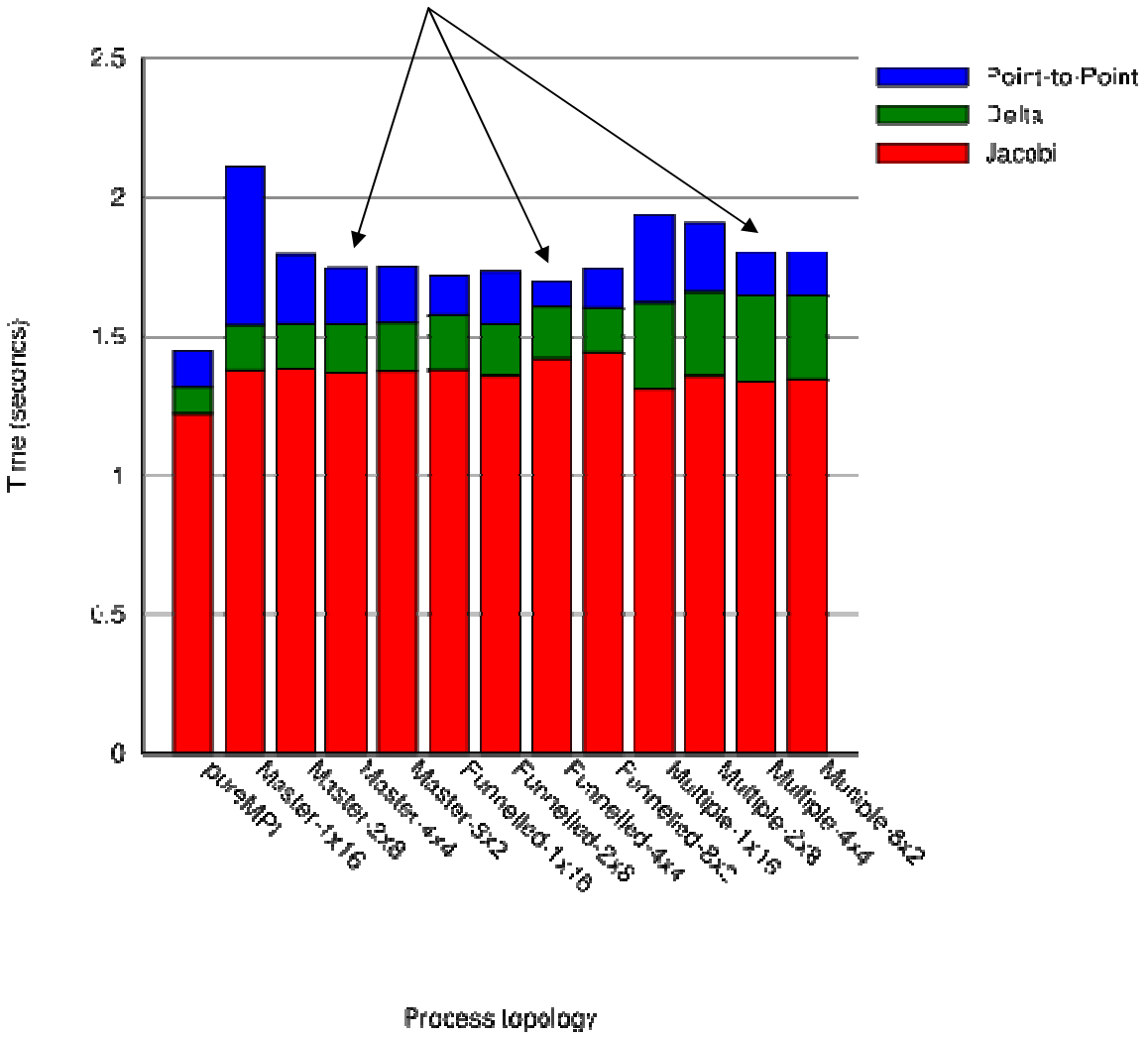


MPI communication

Global Delta

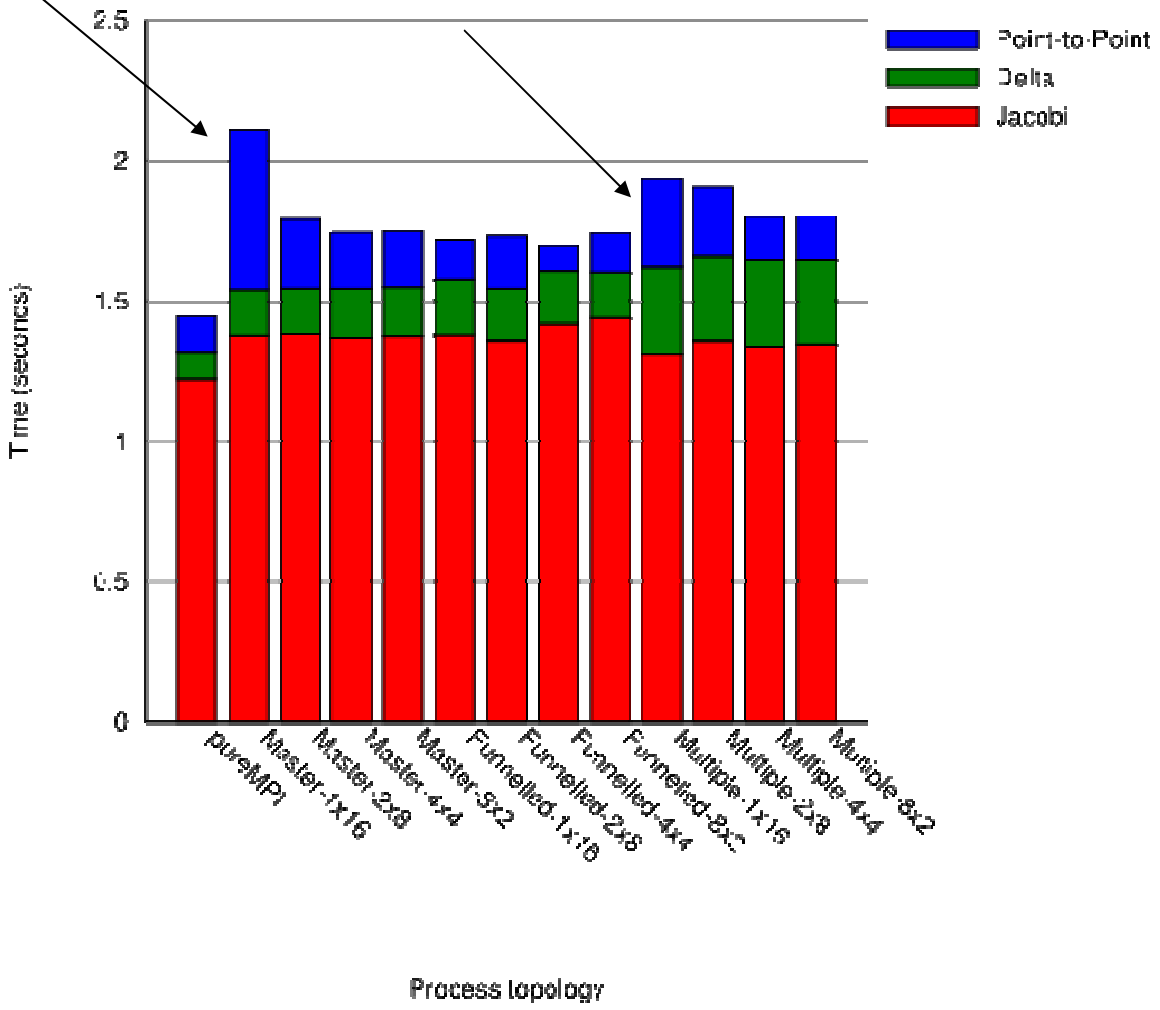
Jacobi relaxation algorithm

4x4 is the most effective topology

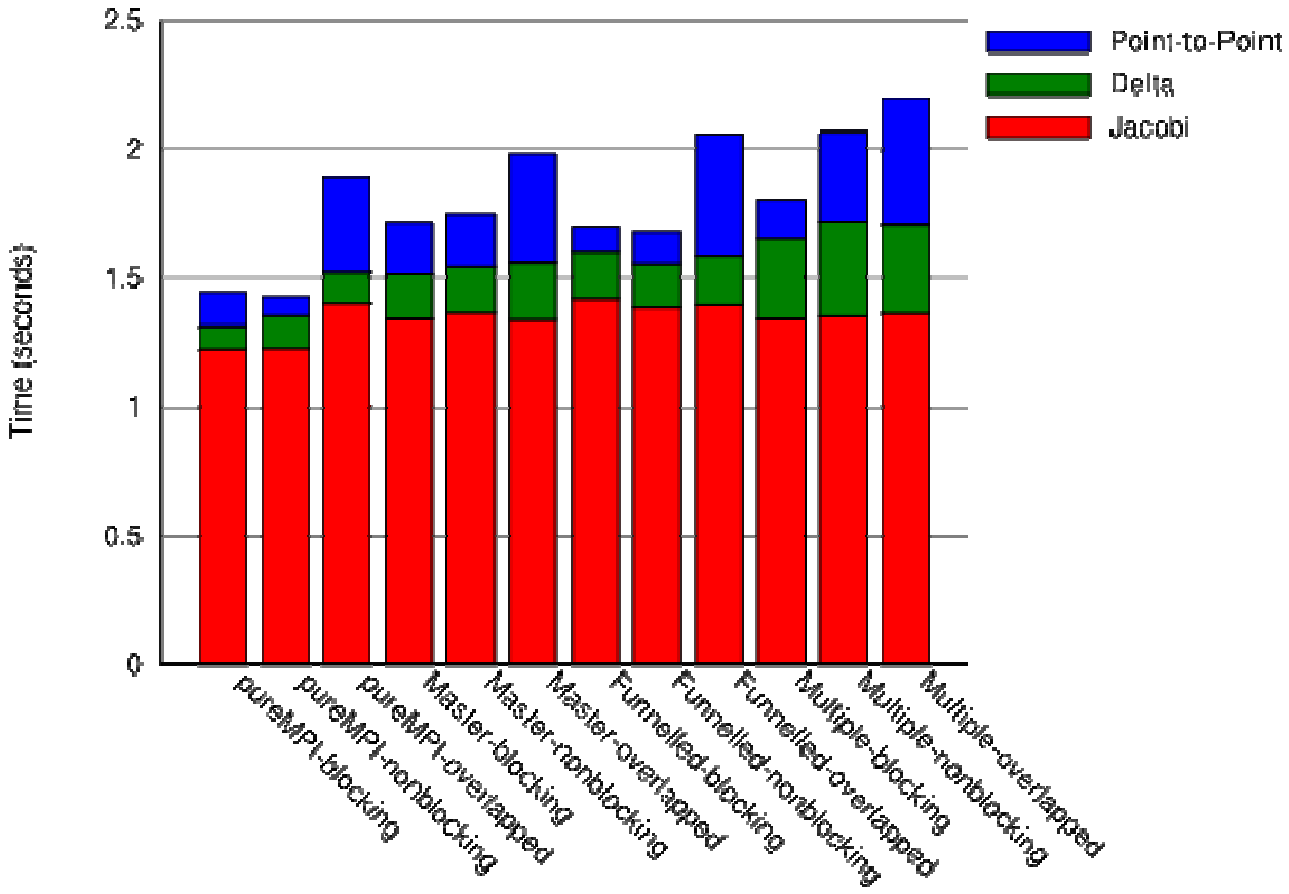


Larger message size

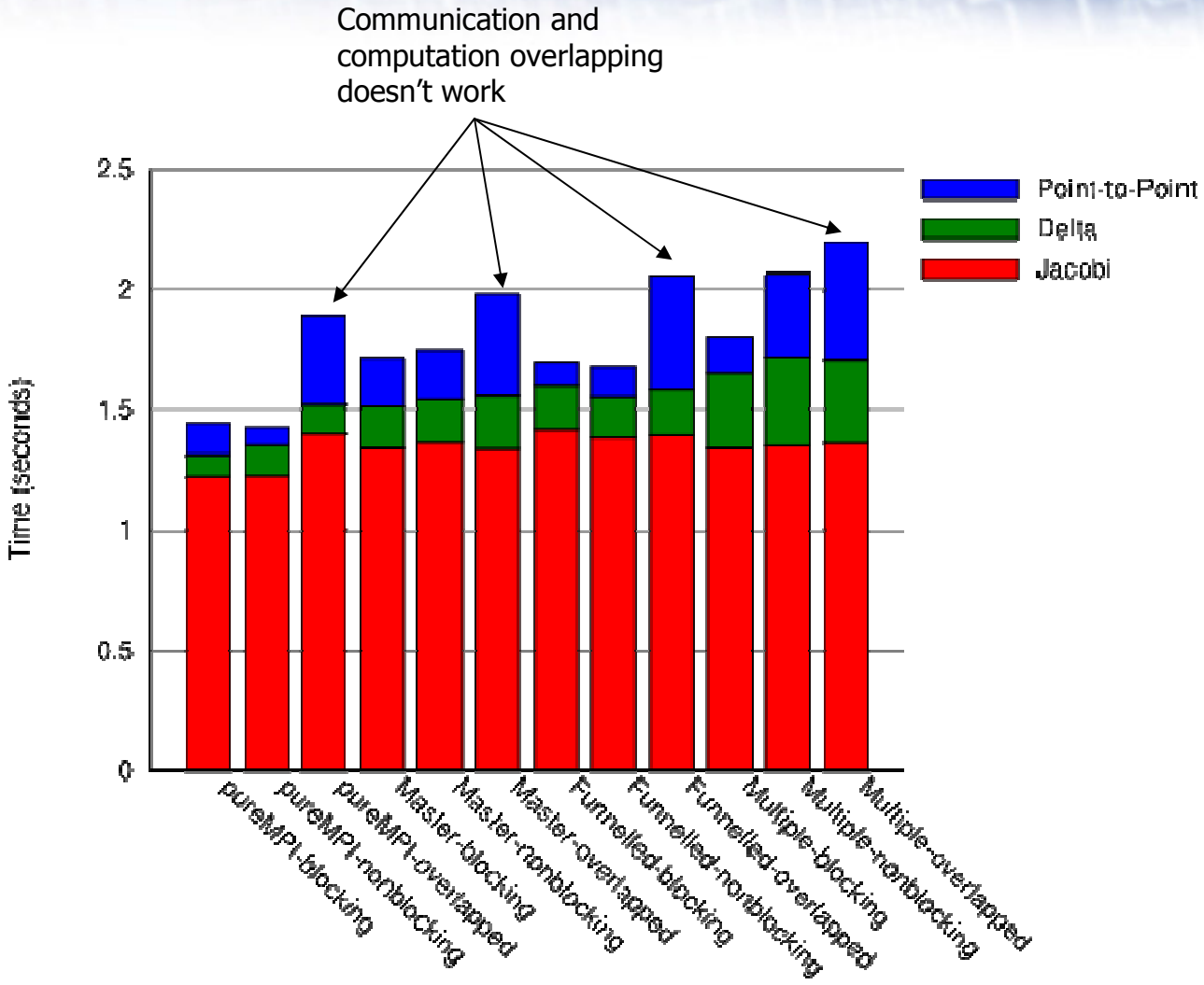
MPI does not handle multiple messages very well



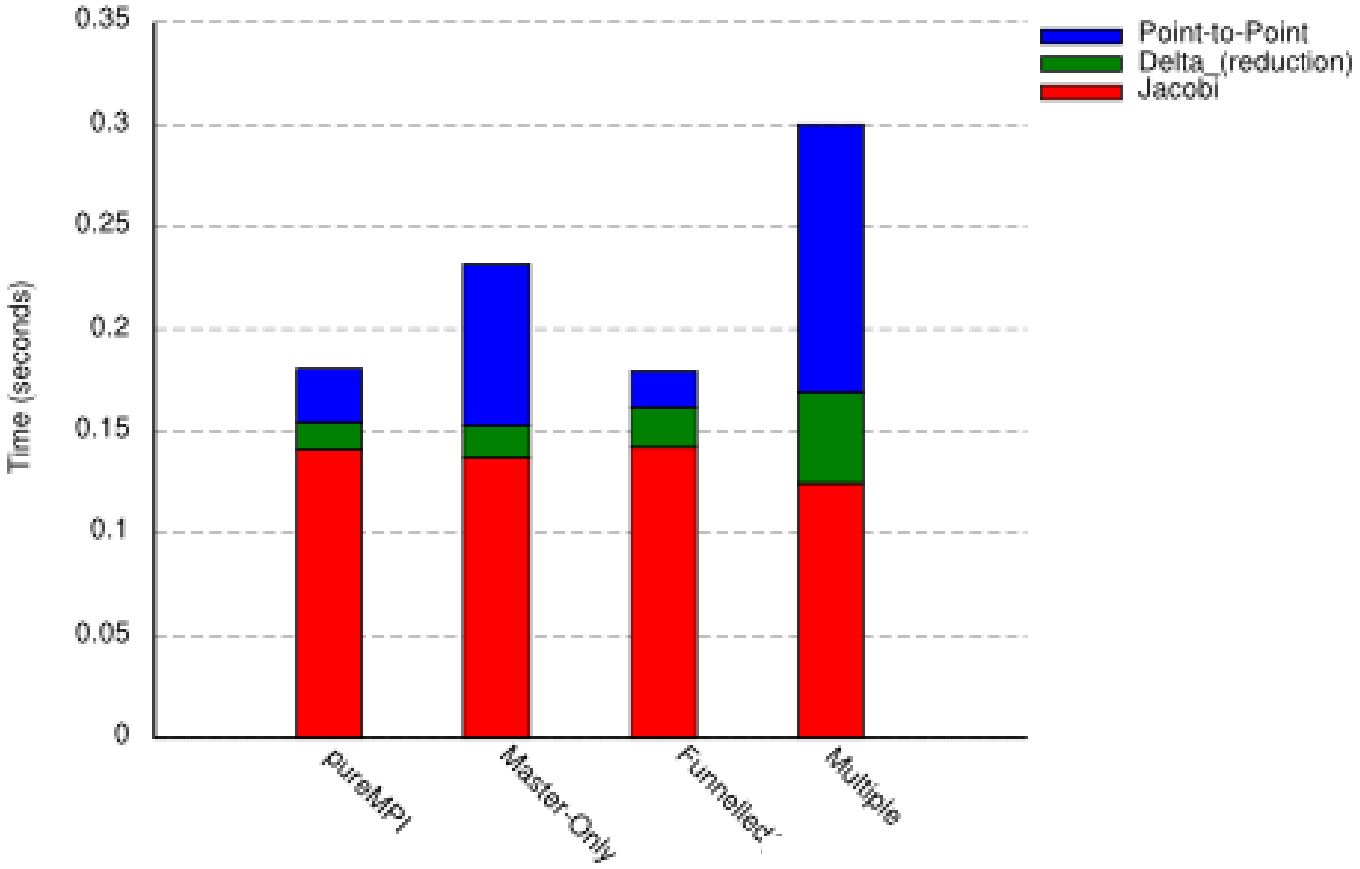
Communication



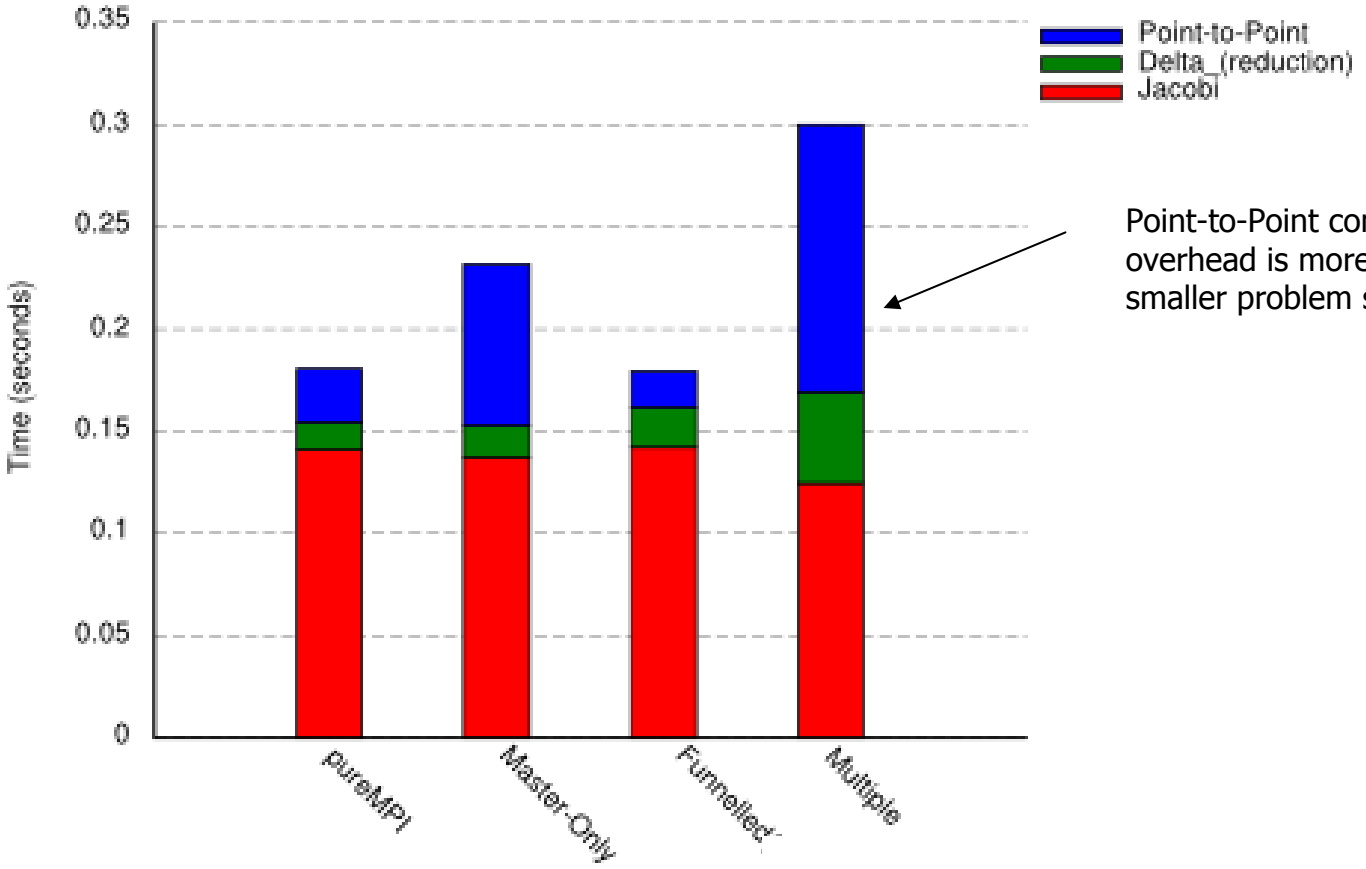
Communication



Basic blocking 96x3

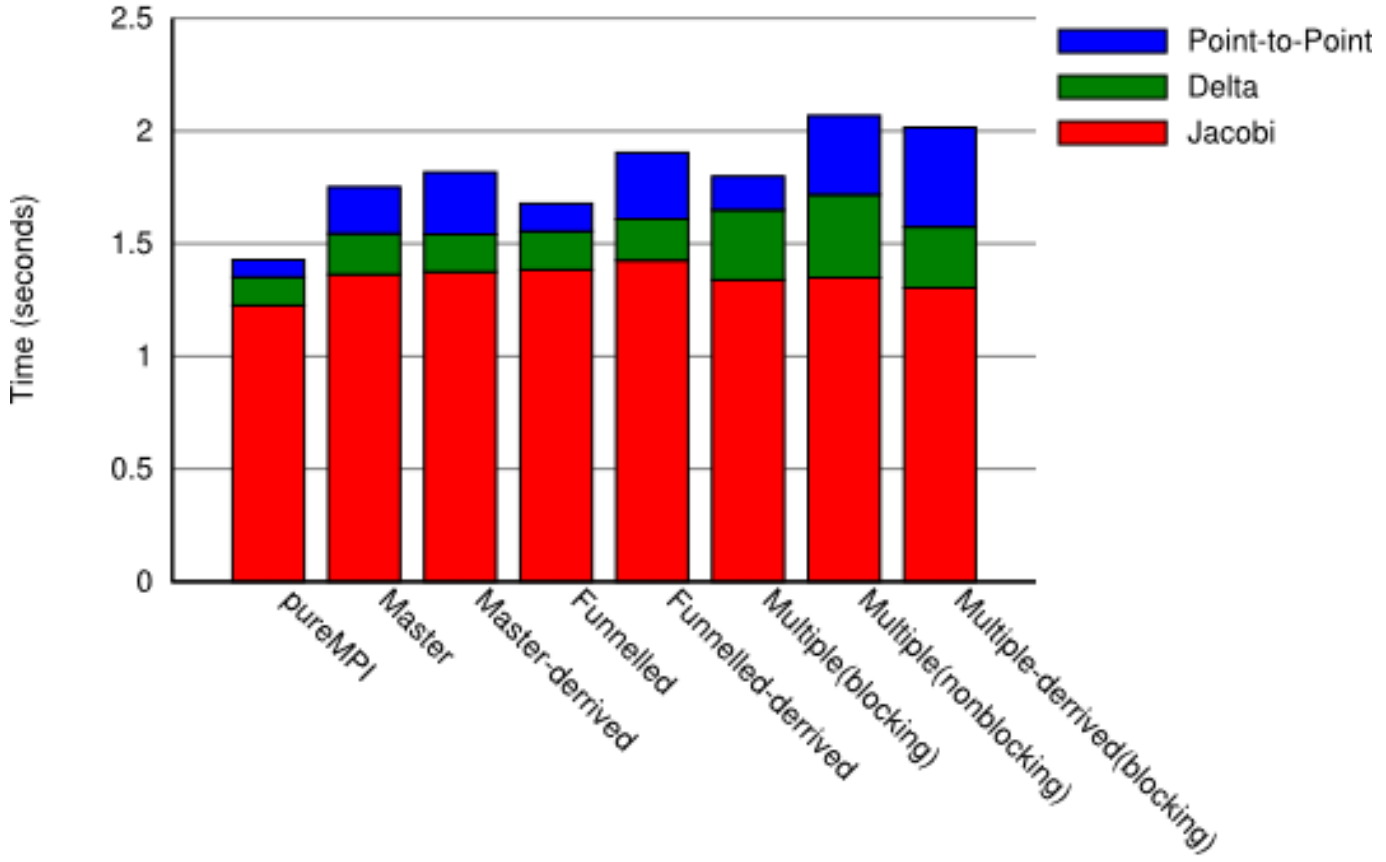


Basic non-blocking 96x3

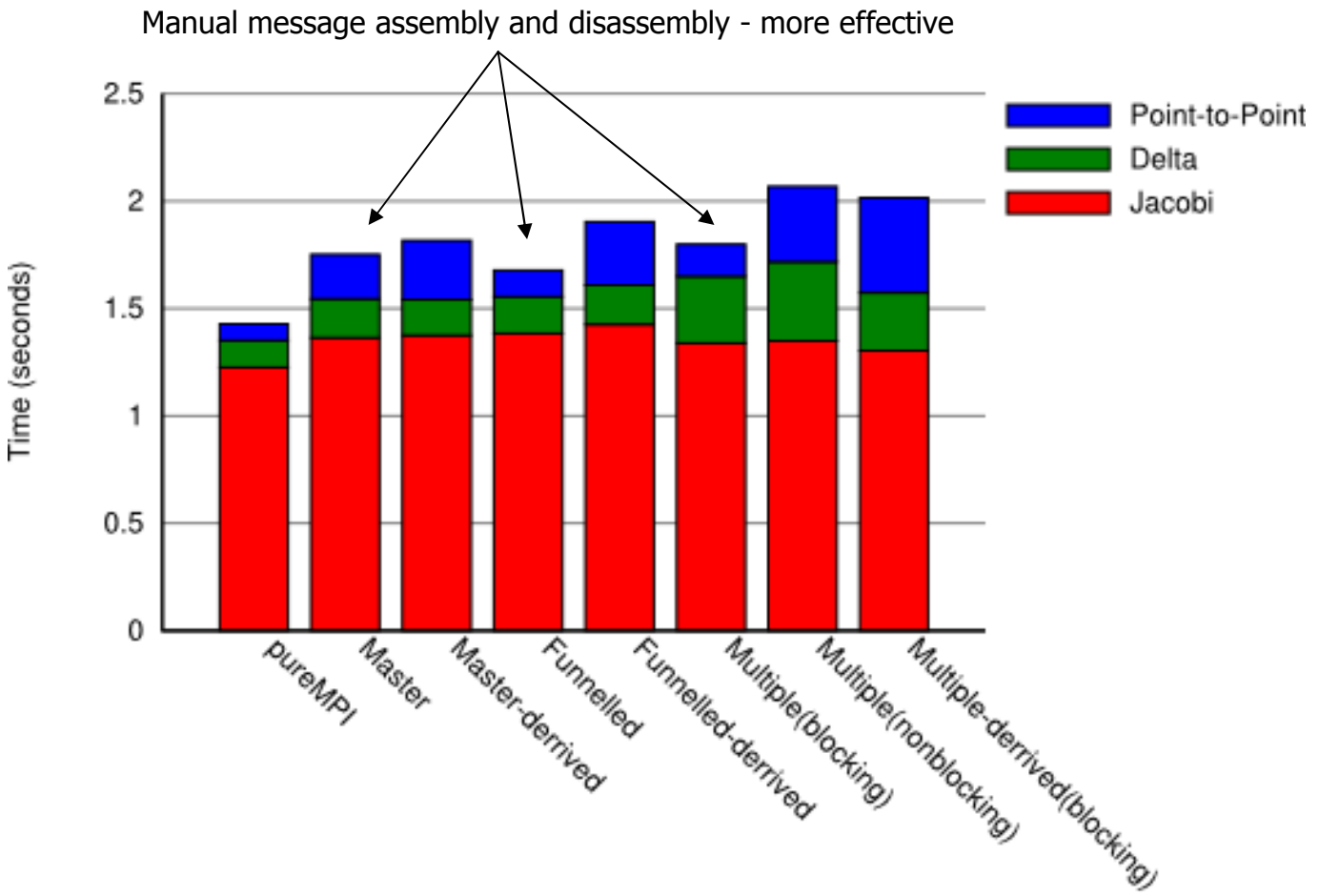


Point-to-Point communication overhead is more visible for smaller problem size.

Derived data types



Derived data types



Conclusions

- None of the mixed mode versions managed to outperform the pure MPI version.
  - Funnelled version 20% slower than pure MPI
  - Ineffective point-to-point communication (Master Only, Funnelled)
  - Limited multi-threading support provided by the MPI library (Multiple)
  - Slow OpenMP Barrier

- There are several scenarios where mixed mode could show its potential:
  - Replicated data problems
  - Some shared memory algorithms may be more efficient
  - Code scales poorly with MPI (Global operations)

Questions